NLO Corrections to High Multiplicity Jet Observables

W. Giele, G. Stavenga and J. Winter

- LO on Graphical Processor Units (GPU's)
- Defining Exclusive Multi-Jet observable at NLO
- NLO on GPU's
  - Real Corrections on the GPU
  - Virtual Corrections on the GPU
  - Color Expansions on the GPU
- Numerics (timing only) and Outlook

Introduction

- Tree-level and one-loop amplitudes can nowadays be calculated up to very high number of external legs
- To combine them in a NLO parton level MC we have to integrate over phase space
- Throw large computer farms at the problem? Or
  - "Redefine the observable": Make the NLO jet phase space identical to the LO jet phase space...
  - Use new and affordable hardware so the MC can run on your own PC again...
- We are pursuing these two options, such that NLO MC's with many jets can run on a single CPU/GPU PC or a multiple CPU/GPU board system...
- For now results are with gluons only and in Leading Color.

LO on GPU's: Hardware

- The *C1060* Nvidia Tesla GPU is a plug-in card for your desktop.
- The Tesla chip is designed for numerical applications and programmable in C/limited C++.
- The chip has *30* multi-processors (MP's), each with *1024* processors (threads).
- For us the limiting factor is the fast on-chip shared memory of *16,384 32*-bit registers per MP (accessible by all threads on the MP).
- (There is *4* Gb off-chip slow access memory we use for I/O only.)

arXiv:1002.3446 [hep-ph]
(vircol.fnal.gov/TESS.html)

$1300 from Amazon.com
Power: 200W @ peak preformance

LO on GPU's: Programming Principles

- All *30x1024=30,720* threads execute the same instructions (threads can skip ahead and wait for other threads to catch-up using if statements etc).
- Each thread has an unique number (for input/output etc)
- A MC generator is trivially parallelizable as the evaluation of each event executes the same instructions using different input (e.g. seeds for the random number generator or momenta).
- So, in principle we can run *30,720* MC generators in parallel, each running *N* events (a speed up of *30,000*!).
- However, fast accessible memory is limiting the number of parallel events.
- Recursion relations perfect for GPU (memory efficient & algorithmic simple). For GPU usage in diagrammatic calculations see Hagiwara, Kanzaki, Okamura, Rainwater, Stelzer (arXiv:0909.5257 [hep-ph], arXiv:0908.4403 [physics.comp-ph])

LO on GPU's: Memory usage

- The *n*-gluon recursion relation needs *n* momenta and *n*(n-1)/2* currents for a total of *n*(n+1)/2* single precision *4*-vectors.
- This means we need *(4*4)*n*(n+1)/2* bytes of fast accesible memory per event.
- The means *16,384/(8*n*(n+1))* events per MP.

n	4	5	6	7	8	9	10	11	12
events/MP	102	68	48	36	28	22	18	15	13
threads/event	10	15	21	28	36	45	55	66	78

Table 1: The number of n-gluon events, which can be simultaneously executed on one MP (and is equal to $2048/[n \times (n+1)]$) and the number of available threads per event (equal to $n \times (n+1)/2$). The total number of events evaluated in parallel on the Tesla chip is $30\times$(events/MP).

LO on GPU's: Timing

AMD Phenom(tm) II X4 940 Processor (3 GHz)

n	$T_n^{\rm GPU}$ (seconds)	$P_n(3)$	$T_n^{\rm CPU}$ (seconds)	$P_n(4)$	G_n
4	2.975×10^{-8}		8.753×10^{-6}		294
5	4.438×10^{-8}	0.91	1.247×10^{-5}	0.87	281
6	8.551×10^{-8}	1.03	1.966×10^{-5}	0.93	230
7	2.304×10^{-7}	1.19	3.047×10^{-5}	0.96	132
8	3.546×10^{-7}	1.01	4.736×10^{-5}	0.98	133
9	4.274×10^{-7}	0.94	7.263×10^{-5}	0.99	170
10	6.817×10^{-7}	1.05	1.044×10^{-4}	0.99	153
11	9.750×10^{-7}	1.02	1.529×10^{-4}	1.00	157
12	1.356×10^{-6}	1.02	2.129×10^{-4}	1.00	158

Table 2: The GPU and CPU evaluation times per event, $T_n^{\rm GPU}$ and $T_n^{\rm CPU}$, given as a function of the number n of gluons for $gg \to (n-2)\,g$ processes. The polynomial scaling measures are also shown, for the GPU, $P_n(3)$, and for the CPU, $P_n(4)$. The $P_n(m)$ are defined as $P_n(m) = [(n-1)/n] \times \sqrt[m]{T_n/T_{n-1}}$. The rightmost column finally displays the gain $G = T_n^{\rm CPU}/T_n^{\rm GPU}$.

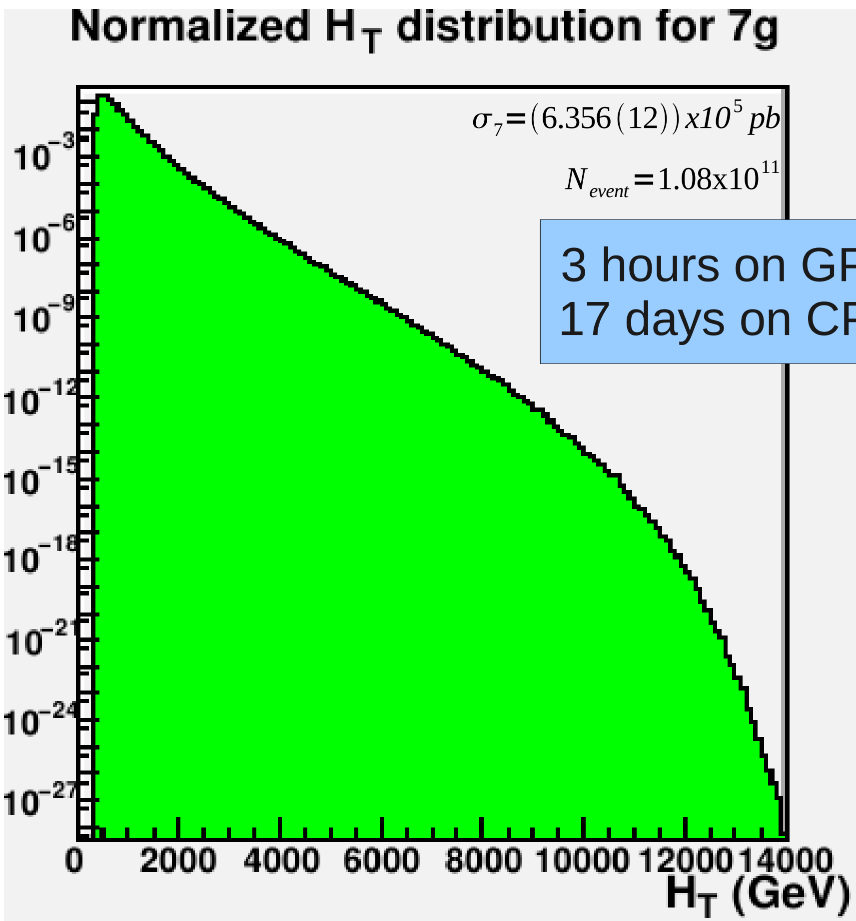
LO on GPU's: Distributions

Kleiss, Stirling, Ellis

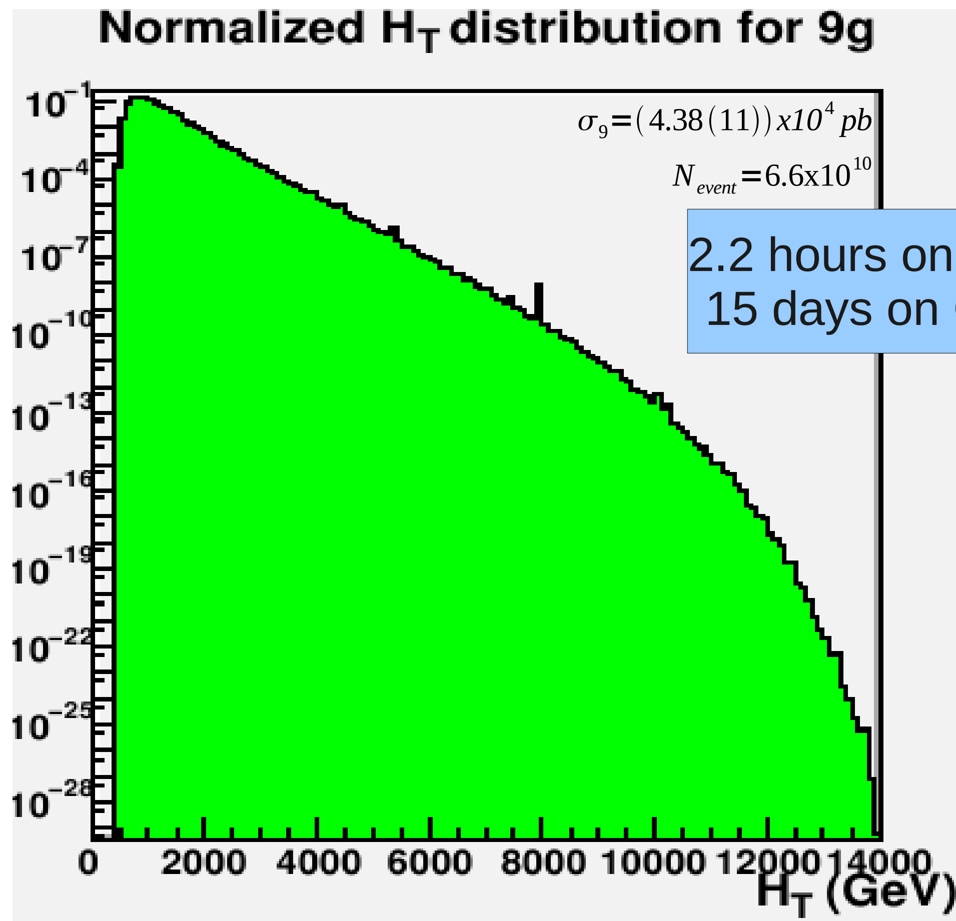
- Rambo generated events for *2g-->5g* and *2g-->7g* in large color limit at *14* TeV (switching to e.g. Haag would dramatically improve statistics).

Van Hameren, Papadopoulos:hep-ph/0204055

- Cuts: *Pt(jet)>60 GeV; |eta(jet)|<2; R(jet,jet)>0.4*



3 hours on GPU
17 days on CPU



2.2 hours on GPU
15 days on CPU

Exclusive Multi-Jet Final States at NLO

- We view perturbative jets as opaque (not as one or two partons).
- That is, we integrate out *all* physics inside the jet cones. This makes the perturbative calculation valid (for appropriate jet cuts).
- The perturbative calculation estimates the correlations between the different jet momenta.
- Making the jet transparent requires for example a shower MC.

LO/NLO jet phase space

- By defining jets in such a way that the NLO and LO jet phase space are identical we achieve:
  - Implicit removal of many potential large logs
  - Each exclusive jet final states have well defined LO and NLO weight (all cancellations occur).
  - Defines “matrix element methods” used by experiments at NLO.
  - Real radiation integral is now *2+1* dimensional. We can go to very high multiplicity jet final states.
- Price to pay is a small augmentation of jet algorithm.

Defining LO/NLO jet phase space

- For final state jets the augmentation is minimal:
  - Apply standard *2->1* clustering according to cut measure *R(p1,p2)*: Massive cluster *p12=p1+p2*
  - Find recoiler *p3* by minimizing *R(p1,p2;p3)* (eg *R(p1,p2;p3)=R(p1,p3)*R(p2,p3)*)
  - Re-scale *p12* and *p3* such that *p12* becomes massless: *p3=a*p3*; *p12=p12-a*p3* (*a=s12/s123*)

LO jet phase space

A jet observable is simply calculated from the exclusive jet cross section:

$$\frac{d\,\sigma}{d\,X} = \int d\,\mathrm{PS}(J_1,\ldots,J_n)\;\delta\left(X - X(J_1,\ldots,J_n)\right)\frac{d^{(n)}\sigma}{d\,J_1\cdots d\,J_n}$$

The LO exclusive jet cross section is given by:

$$\begin{aligned}\frac{d^{(n)}\sigma_{\mathrm{LO}}}{d\,J_1\cdots d\,J_n} &= \int d\,\mathrm{PS}(p_1,\ldots,p_n)\;\delta(J_1-p_1)\cdots\delta(J_n-p_n)\left|\mathcal{M}^{(0)}(p_1,\ldots,p_n)\right|^2\\ &= \left|\mathcal{M}^{(0)}(J_1,\ldots,J_n)\right|^2\;. \end{aligned} \tag{2.2}$$

NLO jet phase space

The NLO exclusive jet cross section is now given by:

$$\frac{d^{(n)}\sigma_{\mathrm{NLO}}}{d\,J_1\cdots d\,J_n} = \int d\,\mathrm{PS}(p_1,\ldots,p_n)\delta(J_1-p_1)\cdots\delta(J_n-p_n)\left|\mathcal{M}^{(0)}(p_1,\ldots,p_n)\right|^2\Big(1+\mathcal{K}(p_1,\ldots,p_n)\Big)$$
$$= \left|\mathcal{M}^{(0)}(J_1,\ldots,J_n)\right|^2\Big(1+\mathcal{K}(J_1,\ldots,J_n)\Big) \qquad (2.6)$$

Each jet phase space point gets a “K-factor” correcting the LO prediction. The “$\mathcal{K}$-factor” is given by

$$\left|\mathcal{M}^{(0)}(p_1,\ldots,p_n)\right|^2\Big(1+\mathcal{K}(p_1,\ldots,p_n)\Big) = \left|\mathcal{M}^{(0)}(p_1,\ldots,p_n)\right|^2 + 2\Re\left(\left[\mathcal{M}^{(0)}\times\mathcal{M}^{(1)^\dagger}\right](p_1,\ldots,p_n)\right)$$
$$+\sum_{a,b}\int d\,\mathrm{PS}(\widehat{p}_a,\widehat{p}_r,\widehat{p}_b|p_a,p_b)\,\theta(\widehat{R}_{arb}<R_{cut})\theta(\widehat{R}_{abr}=\min_{ijk}\widehat{R}_{ijk})\left|\mathcal{M}^{(0)}(\widehat{p}_a,\widehat{p}_r,\widehat{p}_b,p_1,\ldots,p_n)\right|^2 \qquad (2.7)$$

$$d\,\mathrm{PS}(\widehat{p}_a,\widehat{p}_r,\widehat{p}_b|p_a,p_b) = \frac{1}{16\pi^2}\frac{1}{s_{ab}}\,d\,\widehat{s}_{ar}\,d\,\widehat{s}_{rb}\,\theta(\widehat{s}_{ar}>s_{min})\theta(\widehat{s}_{rb}>s_{min})\theta(\widehat{s}_{ar}+\widehat{s}_{rb}<s_{ab}) \qquad (2.8)$$

Only 2-dim integral, independent of # of jets

NLO jet phase space generator

- We now write the *2->3* brancher which is the exact inverse of the jet algorithm.
- This allows us to MC over the real radiation phase space for a *fixed* exclusive jet phase space point.
- Time consuming to evaluate virtual:
  - Generate *M* jet-phase space points with LO weight
  - Un-weight events to give *N<<M* events ← Experimentalists have this already
  - Calculate "K-factor" for un-weighted jet events (should give weights around 1).
  - (E.g. *100,000* LO un-weighted *PP--> 12 jet* events requires *100,000* virtual evaluations,)

Jet Phase Space with Initial State Radiation

Stewart, Tackmann, Waalewijn
arXiv:0910.0467 [hep-ph]

- Use crossing functions (--> beam jets).

Giele, Glover, Kosower: hep-ph/9302225

- The beam jet axis is aligned with the beam particles (not the integrated out internal partons).
- Simple augmentation to jet algorithm:
  - Cluster as described before for final states, including incoming parton.
  - Sometimes partons are cluster with incoming parton -> beam jet.
  - Apply Pt-boost to final jet final state such they are Pt-balanced.
  - We now integrate out automatically the initial state radiation!

(Resummation calculations live in this phase space as it adds radiation in the opaque jet cones without changing the jet kinematics)

NLO timing

Executed on a TESLA C1060 GPU, in parallel with the CPU evaluation of virtual

J. Winter and W. Giele
arXiv:0902.0094 [hep-ph]
On a AMD Phenom(tm) II X4 940 Processor (3 GHz)

# of jets	time/real event (sec)	time/virtual event (sec)	real events/virtual event
2	6.9×10^{-10}	1.6×10^{-2}	6.7×10^{7}
3	2.1×10^{-9}	4.7×10^{-2}	2.2×10^{7}
4	3.7×10^{-9}	1.1×10^{-1}	3.0×10^{7}
5	7.2×10^{-9}	2.4×10^{-1}	3.3×10^{7}
6	1.0×10^{-8}	4.7×10^{-1}	4.7×10^{7}
7	2.0×10^{-8}	8.7×10^{-1}	4.4×10^{7}
8	3.3×10^{-8}	1.5×10^{0}	4.5×10^{7}
9	9.8×10^{-8}	2.7×10^{0}	2.8×10^{7}
10	7.4×10^{-8}	4.5×10^{0}	6.1×10^{7}
11	1.7×10^{-7}	7.5×10^{0}	$4,4 \times 10^{7}$
12	1.9×10^{-7}	1.2×10^{1}	6.4×10^{7}

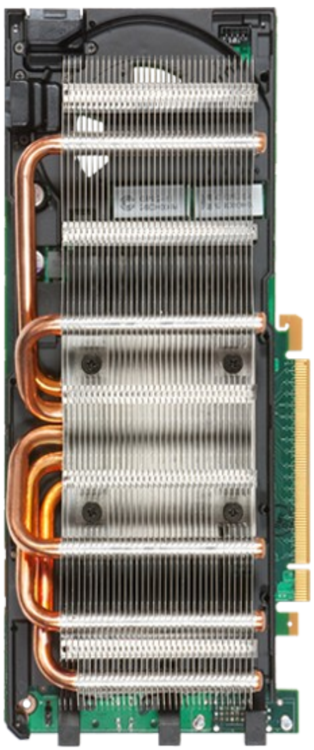
Real diagram ~20x faster than LO diagrams!
(only 3 momenta change/branching).
Linear complexity algorithm for real events!!!

Timing and virtual

- Time is 100% dominated by virtual
- A speed-up with a factor of ~*1000* is desirable
- A factor of *~10* is trivially obtained (let GPU calculate tree-level blobs)
- The next factor of *100* requires a modification in the on-shell methods for GPU implementation
- This is in progress, final GPU implemented virtual should be *100-1000* times faster.
- Current speed: e.g. *~20,000 PP--> 10 jets/day*

Timing and Hardware

- Hardware:
  - Farming with the S2050 GPU rack boards:
    - We run at *16x* the time from the table. (*~300,000 PP-->10 jets/day* at NLO.)
  - New chips come out regularly, the Fermi chip:
    - has *4x* more on-chip memory --> *4x* more events in the same time
    - *7x* double precision speed...
    - Full c++ hardware support--> simplifies coding
    - (almost) doubling # of cores --> *2x* more events
  - New chips will dramatically increase the capabilities for many generations to come.



~2,000,000 pp->10 jets/day

Timing and Outlook

- Given the exclusive definition of jet observables and GPU integration over real phase space we can go to very high multiplicities, eg *2->12*, within reasonable time on a desktop (order of a day or two of running on a GPU). Speeds will further increase dramatically in the coming months/years...
- Next in the line:
  - Virtual on the GPU (in progress)
  - Color-dressed based color expansions on the GPU, keeping polynomial complexity (in progress)
  - Adding quarks in recursion (external and internal)
  - Adding external vector bosons, Higgs,... in recursion